CYO_Dataset

Puja

1/17/2022

OVERVIEW/INTRODUCTION/EXECUTIVE SUMMARY

This data analysis project marks the end of the 9th and final section (Capstone) of the HarvardX Data Science course; made possible by edx. The main objective of this section as a whole is to allow its students to apply all the R skills they've learned thus far in the course in order to see how far they've come.

This project expects similar work from the students as the previous one; with an exception of giving its students the opportunity to choose a data set of their own liking for this one. They are expected to choose a data set, split the data set into a training data set and a testing data set and then familiarize themselves with the training data set by exploring it.

They will ultimately be expected to utilize machine learning skills on the training data set in order to construct different prediction models. This will then be applied to the testing data set in order to see how accurate the different prediction models are.

CHOSEN DATASET

The data set which I decided to choose for my CYO Project is called **"The Chatterjee–Price Attitude Data"** data set. The main reason as to why I decided to choose this specific data set is because it's easily accessible and does not take time to load at all.

The previous project's data set, MovieLens, took extremely long to load and it required you to be connected to the internet in order to access it. I found that to be extremely frustrating & time consuming; and for these reasons, I've decided to go with this data set.

This data set consists of numbers given in percentage proportion taken by a large financial organization. The data was recorded by means of questionnaires of roughly 35 employees from 30 different departments. I will be using my training data set in order to try and predict the rating scores recorded from the questionnaires using my testing data set.

R PACKAGES

1. REQUIRED PACKAGES WHICH NEED TO BE INSTALLED IF THEY HAVE NOT BEEN ALREADY

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dslabs
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(ggpubr)) install.packages("ggpubr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggpubr
```

2. ACCESS TO NECESSARY R PACKAGES

```
library(tidyverse)
library(dslabs)
library(caret)
library(ggpubr)
```

ACCESS TO MY CHOSEN DATASET

The data set which I have chosen to work on for my CYO Project is a build-in data set in R called "attitude". This data set can be easily be accessed by all R users.

OVERALL DATASET SUMMARY

This step shall help broaden our understanding of the chosen dataset

```
head(attitude)
```

```
##   rating complaints privileges learning raises critical advance
## 1     43         51         30       39     61       92      45
## 2     63         64         51       54     63       73      47
## 3     71         70         68       69     76       86      48
## 4     61         63         45       47     54       84      35
## 5     81         78         56       66     71       83      47
## 6     43         55         49       44     54       49      34
```

```
str(attitude)
```

```
## 'data.frame':    30 obs. of  7 variables:
##  $ rating    : num  43 63 71 61 81 43 58 71 72 67 ...
##  $ complaints: num  51 64 70 63 78 55 67 75 82 61 ...
##  $ privileges: num  30 51 68 45 56 49 42 50 72 45 ...
##  $ learning  : num  39 54 69 47 66 44 56 55 67 47 ...
##  $ raises    : num  61 63 76 54 71 54 66 70 71 62 ...
##  $ critical  : num  92 73 86 84 83 49 68 66 83 80 ...
##  $ advance   : num  45 47 48 35 47 34 35 41 31 41 ...
```

```
glimpse(attitude)
```

```
## Rows: 30
## Columns: 7
## $ rating     <dbl> 43, 63, 71, 61, 81, 43, 58, 71, 72, 67, 64, 67, 69, 68, 77,~
## $ complaints <dbl> 51, 64, 70, 63, 78, 55, 67, 75, 82, 61, 53, 60, 62, 83, 77,~
## $ privileges <dbl> 30, 51, 68, 45, 56, 49, 42, 50, 72, 45, 53, 47, 57, 83, 54,~
## $ learning   <dbl> 39, 54, 69, 47, 66, 44, 56, 55, 67, 47, 58, 39, 42, 45, 72,~
## $ raises     <dbl> 61, 63, 76, 54, 71, 54, 66, 70, 71, 62, 58, 59, 55, 59, 79,~
## $ critical   <dbl> 92, 73, 86, 84, 83, 49, 68, 66, 83, 80, 67, 74, 63, 77, 77,~
## $ advance    <dbl> 45, 47, 48, 35, 47, 34, 35, 41, 31, 41, 34, 41, 25, 35, 46,~
```

According to the information above; the data set comprises of 30 rows (observations) and 7 numeric columns (variables) - namely: rating, complaints, privileges, learning, raises, critical and advance.

TRAIN AND TEST DATASET SPLIT

It is of paramount importance that the data which we are busy working with is partitioned into two sets. One will be a training set and the other shall be the testing sets. This is done in order to avoid overfitting and to improve accuracy. The training set and it's observations will first be used to help form the required models. Once the is specific models are formed using the training set, they will then be applied to our test set in order to help analyse the performance of the model.

```
set.seed(33)
cyo_dataset <- attitude
head(cyo_dataset)
```

```
##   rating complaints privileges learning raises critical advance
## 1     43         51         30       39     61       92      45
## 2     63         64         51       54     63       73      47
## 3     71         70         68       69     76       86      48
## 4     61         63         45       47     54       84      35
## 5     81         78         56       66     71       83      47
## 6     43         55         49       44     54       49      34
```

```
dim(cyo_dataset)
```

```
## [1] 30  7
```

For my data split; I have decided to use a **70/30** split. I have chose this ration because I feel as if a 90/10 split would not provide enough observations for my testing set and a 50/50 split won't be able to provide a reading which is accurate enough. Therefore I have decide to go in between with a 70/30 split. I truly believe that this ratio will yield the highest accuracy. I shall now split the data 'cyo_dataset' into two with the help of the 'sample' function.

```
split<- sample(c(rep(1, 0.7 * nrow(cyo_dataset)), rep(2, 0.3 * nrow(cyo_dataset))))
split
```

```
##  [1] 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 2 1 2 2 2 1 1
```

```
table(split)
```

```
## split
##  1  2
## 21  9
```

The data set has now been split 70/30. We have 21 observation in our training set (1) ad 9 observations in our test set (2). They shall now receive their specific names.

TRAINING SET:

```
training_set <- cyo_dataset[split == 1, ]
```

TESTING SET:

```
testing_set <- cyo_dataset[split== 2, ]
```

DATA EXPLORATION

Let us now get an overall summary in order to observe the newly formed training dataset

```
head(training_set)
```

```
##   rating complaints privileges learning raises critical advance
## 1     43         51         30       39     61       92      45
## 2     63         64         51       54     63       73      47
## 4     61         63         45       47     54       84      35
## 5     81         78         56       66     71       83      47
## 7     58         67         42       56     66       68      35
## 8     71         75         50       55     70       66      41
```

```
summary(training_set)
```

```
##      rating        complaints      privileges      learning         raises     
##  Min.   :43.00   Min.   :40.00   Min.   :30.0   Min.   :34.00   Min.   :43.00  
##  1st Qu.:63.00   1st Qu.:61.00   1st Qu.:45.0   1st Qu.:47.00   1st Qu.:59.00  
##  Median :67.00   Median :67.00   Median :50.0   Median :55.00   Median :63.00  
##  Mean   :67.38   Mean   :69.29   Mean   :52.1   Mean   :55.43   Mean   :64.38  
##  3rd Qu.:74.00   3rd Qu.:82.00   3rd Qu.:57.0   3rd Qu.:67.00   3rd Qu.:71.00  
##  Max.   :85.00   Max.   :90.00   Max.   :83.0   Max.   :75.00   Max.   :79.00  
##     critical        advance     
##  Min.   :54.00   Min.   :25.00  
##  1st Qu.:73.00   1st Qu.:35.00  
##  Median :77.00   Median :41.00  
##  Mean   :75.67   Mean   :41.48  
##  3rd Qu.:80.00   3rd Qu.:46.00  
##  Max.   :92.00   Max.   :63.00
```

```
dim(training_set)
```

```
## [1] 21  7
```

```
glimpse(training_set)
```

```
## Rows: 21
## Columns: 7
## $ rating     <dbl> 43, 63, 61, 81, 58, 71, 72, 67, 67, 69, 68, 77, 81, 74, 65,~
## $ complaints <dbl> 51, 64, 63, 78, 67, 75, 82, 61, 60, 62, 83, 77, 90, 85, 60,~
## $ privileges <dbl> 30, 51, 45, 56, 42, 50, 72, 45, 47, 57, 83, 54, 50, 64, 65,~
## $ learning   <dbl> 39, 54, 47, 66, 56, 55, 67, 47, 39, 42, 45, 72, 72, 69, 75,~
## $ raises     <dbl> 61, 63, 54, 71, 66, 70, 71, 62, 59, 55, 59, 79, 60, 79, 55,~
## $ critical   <dbl> 92, 73, 84, 83, 68, 66, 83, 80, 74, 63, 77, 77, 54, 79, 80,~
## $ advance    <dbl> 45, 47, 35, 47, 35, 41, 31, 41, 41, 25, 35, 46, 36, 63, 60,~
```

Distinct numbers belonging to each variable in the training dataset

```
Distinct_Numbers <- training_set %>%
  summarize(D_rating = n_distinct(training_set$rating),
            D_complaints = n_distinct(training_set$complaints),
            D_privileges = n_distinct(training_set$privileges),
            D_learning = n_distinct(training_set$learning),
            D_raises = n_distinct(training_set$raises),
            D_critical = n_distinct(training_set$critical),
            D_advance = n_distinct(training_set$advance))
```

```
Distinct_Numbers
```

```
##   D_rating D_complaints D_privileges D_learning D_raises D_critical D_advance
## 1       17           18           18         18       15         16        14
```

Ratings used the most by employees

```
Popular_Ratings <- training_set %>%
  group_by(rating) %>%
  summarize(num_used = n()) %>%
  top_n(10) %>%
  arrange(desc(num_used))
```

```
## Selecting by num_used
```

```
Popular_Ratings
```

```
## # A tibble: 17 x 2
##    rating num_used
##     <dbl>    <int>
##  1     63        2
##  2     65        2
##  3     67        2
##  4     81        2
##  5     43        1
##  6     50        1
##  7     53        1
##  8     58        1
##  9     61        1
## 10     68        1
## 11     69        1
## 12     71        1
## 13     72        1
## 14     74        1
## 15     77        1
## 16     82        1
## 17     85        1
```

Maximum Value Belonging To Each Variable

```
tab <- matrix(c(max(training_set$rating),
                max(training_set$complaints),
                max(training_set$privileges),
                max(training_set$learning),
                max(training_set$raises),
                max(training_set$critical),
                max(training_set$advance)), ncol=1, byrow = FALSE)
colnames(tab) <- c('Maximum Value in Each Column')
rownames(tab) <- c('Rating', 'Complaints', 'Privileges', 'Learning', 'Raises', 'Critical', 'Advance')
tab <- as.table(tab)
tab
```

```
##            Maximum Value in Each Column
## Rating                               85
## Complaints                           90
## Privileges                           83
## Learning                             75
## Raises                             79
## Critical                           92
## Advance                            63
```

Minimum Value Belonging To Each Variable

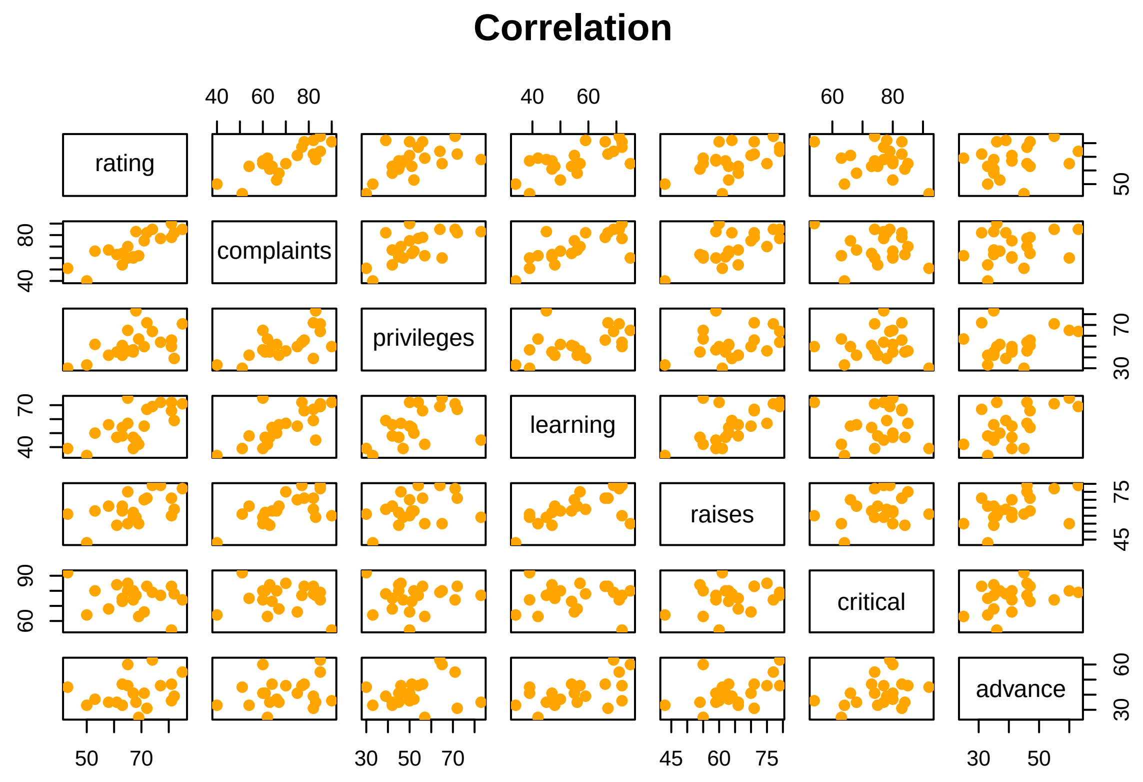
```
tab <- matrix(c(min(training_set$rating),
                min(training_set$complaints),
                min(training_set$privileges),
                min(training_set$learning),
                min(training_set$raises),
                min(training_set$critical),
                min(training_set$advance)), ncol=1, byrow = FALSE)
colnames(tab) <- c('Minimum Value in Each Column')
rownames(tab) <- c('Rating', 'Complaints', 'Privileges', 'Learning', 'Raises', 'Critical', 'Advance')
tab <- as.table(tab)
tab
```

```
##            Minimum Value in Each Column
## Rating                             43
## Complaints                         40
## Privileges                         30
## Learning                           34
## Raises                             43
## Critical                           54
## Advance                            25
```

DATA VISUALIZATION

SCATTER PLOT: CORRELATION BETWEEN RATING AND OTHER VARIABLES

```
plot(training_set, pch = 19, col = c("orange", "orange1"),
     main = "Correlation")
```

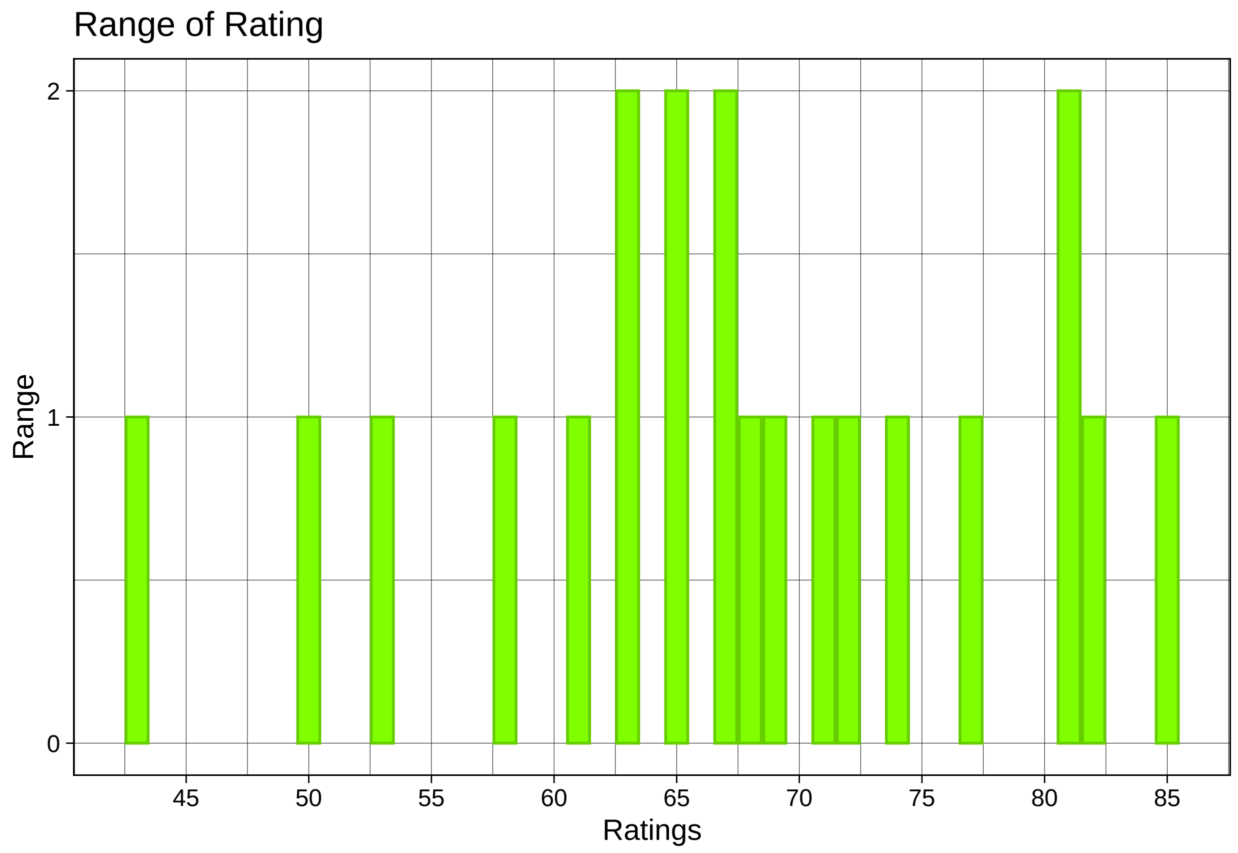


INSIGHT:

According to the scatter plot above, we see that the variables ‘complaints’, ‘raises’ and ‘learning’ have a strong correlation rate with ‘rating’ in that order. In contrast, the variables ‘privileges’, ‘critical’ and ‘advance’ have a poor correlation with ‘rating’.

BAR GRAPH: RANGE OF RATING

```
Range_of_Rating <- training_set %>%
  ggplot(aes(rating)) +
  geom_bar(color = "chartreuse3",  fill = "chartreuse1") +
  scale_x_continuous(breaks = c(35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90)) +
  scale_y_continuous(breaks = c(0, 1, 2, 3)) +
  labs(x = "Ratings", y = "Range") +
  ggtitle("Range of Rating") +
  theme_linedraw()
Range_of_Rating
```

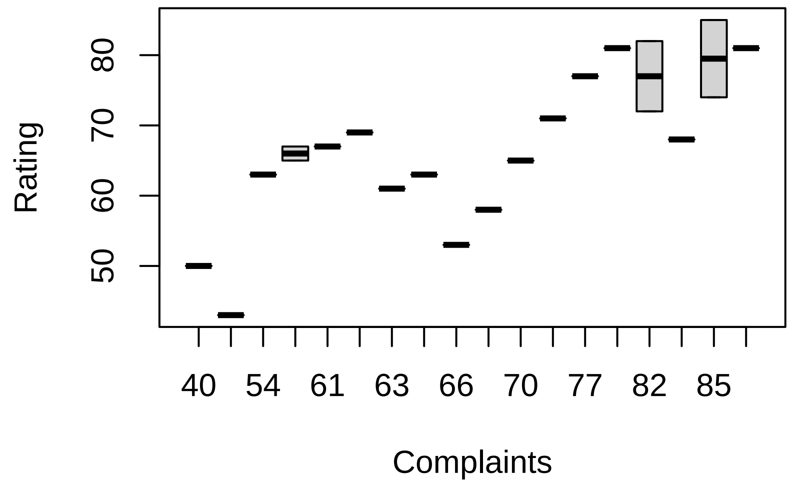


INSIGHTS:

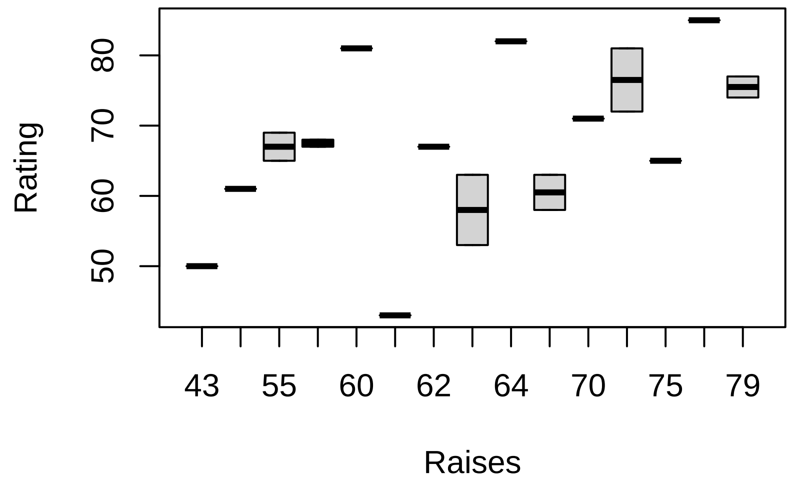
I have chosen to print the ratings that fall in between the numbers 35-90 because according our our data evaluation above; all of the recorded ratings fall within this interval. According to the bar graph shown above, the range of ratings are between 0, 1 and 2 only. Most of the ratings fall between the range of 0 and 1, with the exception of five ratings, which have a range of 2.

BOXPLOT: RANGE OF RATINGS FOR VARIABLES WITH GOOD CORRELATION

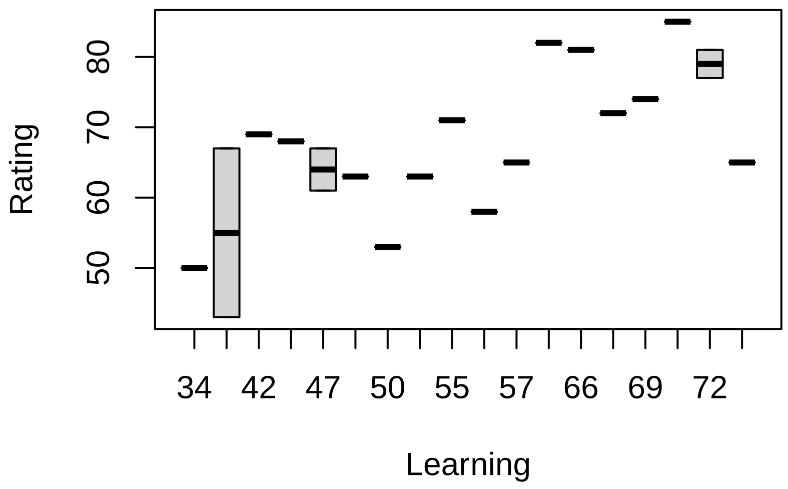
```r
boxplot(training_set$rating~training_set$complaints,
        xlab="Complaints", ylab="Rating")
```



```r
boxplot(training_set$rating~training_set$raises,
        xlab="Raises", ylab="Rating")
```



```r
boxplot(training_set$rating~training_set$learning,
        xlab="Learning", ylab="Rating")
```



INSIGHTS:

BOXPLOT 1 - This boxplot seems to show a dip in the middle of the plot, and then a sharp increase towards the right. This shows us that an increase in resolved complaints leads to an increase in ratings.

BOXPLOT 2 - This boxplot seems to be relatively even throughout. This shows us that departments which offer their employees raises receive higher ratings compared to departments which do not.

BOXPLOT 3 - This boxplot shows us that there is a positive relationship between learning and ratings; however, most of the ratings fall towards the lower left corner of the plot.

MODELING OF DATA

After analyzing all of the data evaluation and data visualization results above, we can now say that we have a very clear idea of the data set with which we are working on. We shall now use the knowledge gained thus far in order to construct models using the training set and implement them onto the testing set to predict the accuracy of the models.

We shall measure the accuracy of the models through the use of the Residual Mean Squared Error (RMSE) Formula. This formula assists in measuring the error between the observed and the predicted values.

RMSE Formula

```
RMSE_Formula <- function(observed, predicted){
  sqrt(mean((observed - predicted)^2))
}
```

I shall now represent "testing_set$rating" as "b" in order to save myself from typing it out whenever I have to calculate the RMSE.

```
b <- testing_set$rating
```

1ST MODEL

The first model which I will be implementing onto the training set, and eventually onto the testing set is the most simplest of them all. This shall be done by using the mean of the ratings only.

Calculate the mean/average of all recorded ratings.

```
mu_cap <- mean(training_set$rating)
```

RMSE of Model 1

```
RMSE_MODEL_1 <- RMSE_Formula(b, mu_cap)
RMSE_MODEL_1
```

```
## [1] 15.53668
```

INSIGHTS GAINED FROM MODEL 1: The resulting RMSE for Model 1 is extremely high. Much higher than I had expected it to be. I shall now try to reduce it in the next model.

2ND MODEL

I shall now attempt to reduce the previously shown RMSE result by upgrading my first model. I shall try doing this by including the variable 'complaints'.

Measure the effect of all complaints to the data set. Compute penalty term c1; associated with complaints

```
complaints_rating <- training_set %>%
  group_by(complaints) %>%
  summarise(c1 = mean(rating - mu_cap))
complaints_rating_forecast <- mu_cap + testing_set %>%
  left_join(complaints_rating, by = 'complaints') %>%
  .$c1
```

RMSE OF MODEL 2

```
RMSE_MODEL_2 <- RMSE_Formula(b, complaints_rating_forecast)
RMSE_MODEL_2
```

```
## [1] NA
```

INSIGHTS GAINED FROM MODEL 2: I am extremely disheartened to see that the RMSE for model 2 has a result of N/A. Unfortunately, I have no idea as to why this has happened. This means that I will have to find another way to reduce the RMSE value.

3RD MODEL

I shall now implement the **Regulization** method to the data set in the hopes of obtaining a lower RMSE reading than the first model. Unfortunately, the second model gave a result of N/A. Hopefully this method will give a more promising RMSE result.

Regulization

```
Regulization_Lmd <- seq(0, 15, 0.5)

Regulization_RMSE <- sapply(Regulization_Lmd, function(lmd){
    Regulization_c1 <- training_set %>%
      group_by(complaints) %>%
      summarize(Regulization_c1 = sum(rating - mu_cap)/(n() + lmd))

Regulization_c2 <- training_set %>%
      left_join(Regulization_c1, by = "complaints") %>%
      group_by(raises) %>%
      summarize(Regulization_c2 = sum(rating - Regulization_c1 - mu_cap)/(n() + lmd))

Regulization_c1_c2_mu_Forecast <- testing_set %>%
      left_join(Regulization_c1, by = "complaints") %>%
      left_join(Regulization_c2, by = "raises") %>%
      mutate(forecast_reg = mu_cap + Regulization_c1 + Regulization_c2) %>%
      pull(forecast_reg)
    return(RMSE_Formula(b, Regulization_c1_c2_mu_Forecast))
  })
```

RMSE OF MODEL 3

```
RMSE_MODEL_3 = min(Regulization_RMSE)
RMSE_MODEL_3
```

```
## [1] NA
```

INSIGHTS GAINED FROM MODEL 3: Sadly, the RMSE reading has yet again come up as N/A after using regulization wih the variables ‘complaints’ and ‘raises’. This means that I will have to try another method in order to get a lower RMSE than the first model.

4TH MODEL

I shall now try a different approach and use the standard deviation to try and get a lower RMSE reading compared to the first model, which made use of the mean.

```
sd_cap <- sd(training_set$rating)
```

RMSE of Model 4

```
RMSE_MODEL_4 <- RMSE_Formula(b, sd_cap)
RMSE_MODEL_4
```

```
## [1] 49.01426
```

INSIGHTS GAINED FROM MODEL 4: On the bright side, this model managed to produce a RMSE reading with digits and not 'N/A'. However, this RMSE reading is not lower than the first model. This means that another method will have to be used in order to reduce the RMSE value.

5TH MODEL

Seeing that our previously constructed models were regrettably unable to achieve a strong RMSE I have now decided to use the only other method which comes to mind: the **Matrix Factorization** method. I shall now implement this method in the hopes of obtaining a lower RMSE value compard to the first constructed model.

MATRIX FACTORIZATION

```
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: recosystem
```

```
library(recosystem)
set.seed(1)
matrifac_r <- Reco()
matrifactest <- with(testing_set, data_memory(rating = rating, user_index = complaints, item_index = ra
matrifactrain <- with(training_set, data_memory(rating = rating, user_index = complaints, item_index = 

matrifac_p <- matrifac_r$tune(matrifactrain, opts = list(niter = 10, nthread = 1, dim = c(1, 20)))
matrifac_r$train(matrifactrain, opts = c(matrifac_p$min, niter = 10, nthread = 1))
```

```
## iter      tr_rmse          obj
##    0      64.0090   8.6079e+04
##    1      54.4245   6.2268e+04
##    2      44.9873   4.2587e+04
##    3      37.4117   2.9495e+04
##    4      31.2962   2.0685e+04
##    5      26.3605   1.4720e+04
##    6      22.4501   1.0720e+04
##    7      19.3219   7.9828e+03
##    8      16.7451   6.0368e+03
##    9      14.5522   4.6002e+03
```

```
matrifac_answer <- matrifac_r$predict(matrifactest, out_memory())
```

RMSE OF MODEL 5

```
RMSE_MODEL_5 <- RMSE_Formula(b, matrifac_answer)
RMSE_MODEL_5
```

```
## [1] 15.51108
```

INSIGHTS GAINED FROM MODEL 5: This is great! I have finally succeeded in obtaining an RMSE value which is lower than all previous results. I am extremely pleased with this result.

RESULTS

I shall now create a 'results' table in which I shall record all of the RMSE Model recordings.

```r
tab <- matrix(c(RMSE_MODEL_1, RMSE_MODEL_2, RMSE_MODEL_3, RMSE_MODEL_4, RMSE_MODEL_5),
              ncol=1, byrow = FALSE)
colnames(tab) <- c('RMSE Results')
rownames(tab) <- c('RMSE MODEL 1', 'RMSE MODEL 2', 'RMSE MODEL 3', 'RMSE MODEL 4',
                   'RMSE MODEL 5')
tab <- as.table(tab)
tab
```

```
##              RMSE Results
## RMSE MODEL 1     15.53668
## RMSE MODEL 2
## RMSE MODEL 3
## RMSE MODEL 4     49.01426
## RMSE MODEL 5     15.51108
```

I am extremely happy that I managed to achieve an RMSE result that is lower than the one previously obtained. Despite this triumph; I will admit that I was hoping for a much lower RMSE result, but I find peace in knowing that I tried my best in trying to do so.

CONCLUSION

The 'Choose Your Own' Project was a great way to end this marvelous program! If I could change anything regarding my project, I think that I'd most probably choose a different dataset than the one I chose. Nevertheless, I managed to obtain a decent RMSE result which I am happy with. I'm glad I was able to better my previous result.

I am extremely grateful to edx and HarvardX for creating such an amazing data science program. After completing all 9 courses, I feel much more confident when it comes to coding in R and programming in general. I've managed to learn so much and it amazes me as to wide the scope of coding can go! It's extremely limitless! You can literally code anything! This was an extremely joyful and insightful experience.